Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)

Towards Feasible Higher-Dimensional Potential Heuristics

Daniel Fišer[1], **Marcel Steinmetz**[2]

[1]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2]LAAS-CNRS, ANITI, Université de Toulouse, Toulouse, France
danfis@danfis.cz, marcel.steinmetz@laas.fr

Abstract

Potential heuristics assign numerical values (potentials) to state features, where each feature is a conjunction of facts. It was previously shown that the informativeness of potential heuristics can be significantly improved by considering complex features, but computing potentials over all pairs of facts is already too costly in practice. In this paper, we investigate whether using just a few high-dimensional features instead of all conjunctions up to a dimension n can result in improved heuristics while keeping the computational cost at bay. We focus on (a) establishing a framework for studying this kind of potential heuristics, and (b) whether it is reasonable to expect improvement with just a few conjunctions. For (a), we propose two compilations that encode each conjunction explicitly as a new fact so that we can compute potentials over conjunctions in the original task as one-dimensional potentials in the compilation. Regarding (b), we provide evidence that informativeness of potential heuristics can be significantly increased with a small set of conjunctions, and these improvements have positive impact on the number of solved tasks.

1 Introduction

In classical optimal planning, potential heuristics (Pommerening et al. 2015; Pommerening, Helmert, and Bonet 2017) are a family of admissible (and consistent) heuristics computed as sums of potentials (numerical values) over state features that are sets of facts (conjunctions). Potentials ensuring admissibility and consistency can be found by solving a certain linear program with any optimization criteria that can be designed to emphasize different properties of the resulting heuristics (Seipp, Pommerening, and Helmert 2015; Fišer, Horčík, and Komenda 2020). It was shown that increasing the size of conjunctions can eventually lead to optimal heuristics (Pommerening, Helmert, and Bonet 2017). Computation of potential heuristics over single facts and pairs of facts is polynomial. However, computing the heuristics over all pairs of facts is already too computationally costly in practice, and for conjunctions of size 3 or more, it becomes coNP-hard to decide heuristic consistency.

Here, we focus on the question whether computing potential heuristics over just a few conjunctions can lead to a significantly more informed heuristic. Corrêa and Pommerening (2019) partially addressed this question in their study of the optimal correlation complexity of planning tasks expressed as the minimum conjunction size for which there exists a potential heuristic equal to the optimal heuristic. They provided some evidence that a small number of large conjunctions is often enough to significantly increase informativeness of potential heuristics. However, their analysis requires full expansion of the reachable part of the state space and therefore it is impractical. We take one step further by showing how to compute potential heuristics over a set of conjunctions so that it can be used in practice.

We leverage prior work of Steinmetz and Hoffmann (2018) showing that potential heuristics over conjunctions can be computed via the so-called $\Pi^{\mathcal{C}}$ compilation (Haslum 2012) where conjunctions $\mathcal{C}$ are explicitly represented as facts. Potential heuristics over conjunctions $\mathcal{C}$ can be computed as atomic (single-fact) potential heuristics in $\Pi^{\mathcal{C}}$, but the price we pay is the worst-case exponential blow-up in $|\mathcal{C}|$ of the $\Pi^{\mathcal{C}}$ encoding. Since $\Pi^{\mathcal{C}}$ does not preserve state space of the original task perfectly and thus does not preserve all (potential) heuristics, we introduce a new compilation $\Pi^{\mathcal{C}}_{\text{exact}}$ that remedies this pitfall. This allows us to plug-in a set of conjunctions $\mathcal{C}$ as input, and obtain a potential heuristic over $\mathcal{C}$ as output, while exchanging computational complexity for blow-up in the task encoding. Moreover, we use prior work on mutual exclusion state invariants to mitigate the blow-up (Keyder, Hoffmann, and Haslum 2014; Fišer and Komenda 2018; Fišer, Horčík, and Komenda 2020).

To test whether it is possible to increase informativeness of potential heuristics with only few conjunctions, we use a simple greedy uninformed algorithm to obtain improving conjunctions. We show in our experiments that it is, indeed, often the case that a small number of conjunctions leads to better heuristic estimates. We test this approach with explicit-state search, but also with symbolic search where potential heuristics can be used via so-called operator-potential heuristics (Fišer, Torralba, and Hoffmann 2022a,b). We show that even in this simple setting, we are able to increase the number of solved tasks in some domains. We leave the question how to intelligently find the improving conjunctions to future work.

2 Background

An **FDR planning task** (Bäckström and Nebel 1995) is a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$. $\mathcal{V}$ is a finite set of **variables**, each

$v \in \mathcal{V}$ has a finite **domain** $\mathrm{dom}(v)$. A **fact** $\langle v, x\rangle$ is a pair of a variable $v \in \mathcal{V}$ and one of its values $x \in \mathrm{dom}(v)$. The set of all facts is denoted by $\mathcal{F} = \{\langle v, x\rangle \mid v \in \mathcal{V}, x \in \mathrm{dom}(v)\}$, the set of facts of variable v is denoted by $\mathcal{F}_v = \{\langle v, x\rangle \mid x \in \mathrm{dom}(v)\}$, and similarly for sets of variables V: $\mathcal{F}_V = \bigcup_{v \in V} \mathcal{F}_v$. Given $p \subseteq \mathcal{F}$, $\mathcal{V}(p)$ denotes all variables appearing in p, i.e., $\mathcal{V}(p) = \{v \mid \langle v, x\rangle \in p\}$, and we use a shorthand $\mathcal{F}_p = \mathcal{F}_{\mathcal{V}(p)}$.

A **partial state** $p \subseteq \mathcal{F}$ is a set of facts s.t. there is at most one fact of each variable, i.e., $|p \cap \mathcal{F}_v| \leq 1$ for every $v \in \mathcal{V}$. $p[v]$ denotes the value assigned to $v \in \mathcal{V}(p)$ in the partial state p. A partial state s is called **state** if $|s| = |\mathcal{V}|$. I is an **initial state**. G is a partial state called **goal**, and a state s is a **goal state** if $G \subseteq s$. A partial state $p \subseteq \mathcal{F}$ is also called a **conjunction**, and we say that a conjunction c is true in the state s if $c \subseteq s$ and we say it is false otherwise. Let p, t be partial states. We say that t **extends** p if $p \subseteq t$.

$\mathcal{O}$ is a finite set of **operators**, $o \in \mathcal{O}$ is defined by its partial states precondition $\mathrm{pre}(o)$ and effect $\mathrm{eff}(o)$, and a cost $\mathrm{cost}(o) \in \mathbb{R}_0^+$. We assume $\mathrm{pre}(o) \cap \mathrm{eff}(o) = \emptyset$. $o \in \mathcal{O}$ is **applicable** in a state s if $\mathrm{pre}(o) \subseteq s$. The **resulting state** of this application is $o[\![s]\!] = (s \setminus \mathcal{F}_{\mathrm{eff}(o)}) \cup \mathrm{eff}(o)$, i.e., applying o on s changes the values of variables according to the effect $\mathrm{eff}(o)$ and keeps the variables not mentioned in $\mathrm{eff}(o)$ unaffected. We also use $\mathrm{post}(o) = (\mathrm{pre}(o) \setminus \mathcal{F}_{\mathrm{eff}(o)}) \cup \mathrm{eff}(o)$ to denote the post-condition of o, i.e., the partial state that is true after every application of o.

A sequence of operators $\pi = \langle o_1, \ldots, o_n\rangle$ is applicable in a state s_0 if there are states $s_1, \ldots, s_n$ such that o_i is applicable in s_{i-1} and $s_i = o_i[\![s_{i-1}]\!]$ for $i \in \{1, \ldots, n\}$. The resulting state is $\pi[\![s_0]\!] = s_n$ and $\mathrm{cost}(\pi) = \sum_{i=1}^{n} \mathrm{cost}(o_i)$ denotes the cost of π. A sequence of operators π is called an **s-plan** if π is applicable in the state s and $\pi[\![s]\!]$ is a goal state. I-plans are simply called **plans**. An s-plan π is called **optimal** if its cost is minimal among all s-plans. A sequence of operators applicable in the initial state is called a **path**.

A state s is called **reachable** if there exists a path π such that $\pi[\![I]\!] = s$. $\mathcal{R}(\Pi)$ denotes the set of all reachable states in Π. An operator o is **reachable** if it is applicable in some reachable state. A state s is a **dead-end** if $G \not\subseteq s$ and there is no s-plan. A **heuristic** is a function $h : \mathcal{R}(\Pi) \mapsto \mathbb{R} \cup \{\infty\}$ estimating the cost of optimal s-plans. The **optimal heuristic** $h^\star(s)$ maps each reachable state s to the cost of the optimal s-plan or to ∞ if s is a dead-end. A heuristic h is called (a) **forward admissible** (f-admissible) if $h(s) \leq h^\star(s)$ for every $s \in \mathcal{R}(\Pi)$; (b) **forward goal-aware** (f-goal-aware) if $h(s) \leq 0$ for every reachable goal state s; and (c) **forward consistent** (f-consistent) if $h(s) \leq h(o[\![s]\!]) + \mathrm{cost}(o)$ for all $s \in \mathcal{R}(\Pi)$ and $o \in \mathcal{O}$ applicable in s.

Note that we define heuristics over the reachable states (instead of all states) because we intend to use heuristics in a forward heuristic search and because we use state invariants describing the reachable state space for improving the heuristic values (h-values). Also note that we allow negative heuristic values as is usual in literature on potential heuristics (the standard interpretation is that during the search, negative heuristic values are interpreted as zero). It is well-known that (forward) goal-aware and (forward) consistent heuristics are also (forward) admissible.

3 Mutexes and Disambiguation

A mutex is a set of facts that is not part of any reachable state, i.e., $M \subseteq \mathcal{F}$ is a **mutex** if $M \not\subseteq s$ for every $s \in \mathcal{R}(\Pi)$. Fišer, Horčík, and Komenda (2020) shown that using mutexes can significantly improve informativeness of potential heuristics. The most obvious mutex in an FDR task is a pair of facts of the same variable, but more mutexes can be inferred by the h^m heuristic (Bonet and Geffner 2001; Alcázar and Torralba 2015) or by inference of so called fam-groups (Helmert 2009; Fišer 2020, 2023; Fišer and Komenda 2018). Clearly, every superset of a mutex is also a mutex. For notational convenience, we use the notion of a mutex-set.

Definition 1. A set of sets of facts $\mathcal{M} \subseteq 2^{\mathcal{F}}$ is called a **mutex-set** if (a) every $M \in \mathcal{M}$ is a mutex, and (b) for every $M \in \mathcal{M}$ and every $f \in \mathcal{F}$ it holds that $M \cup \{f\} \in \mathcal{M}$, and (c) for every variable $v \in \mathcal{V}$ and every pair of facts $f, f' \in \mathcal{F}_v$, $f \neq f'$, it holds that $\{f, f'\} \in \mathcal{M}$.

In other words, a mutex-set is an upper set of a set of mutexes (a,b) and it always contains all mutexes that can be inferred directly from task's variables (c). This allows us to write $p \in \mathcal{M}$ if we want to express that the set of facts p is not a partial state (i.e., it contains two facts of the same variable), or that all states extending p are not reachable. Note also that $p \notin \mathcal{M}$ implies p is a partial state.

Mutexes can also be used for inferring disambiguations (Fišer, Horčík, and Komenda 2020). A disambiguation of a variable v for a partial state p is a set of facts $X \subseteq \mathcal{F}_v$ from the same variable v such that every reachable state extending p contains a fact from X. In other words, disambiguation of v for p allows us to filter out facts of the variable v that cannot be part of any reachable state extending p.

Definition 2. Let $v \in \mathcal{V}$ denote a variable, and let p denote a partial state. $X \subseteq \mathcal{F}_v$ is called a **disambiguation of v for** p if for every $s \in \mathcal{R}(\Pi)$ s.t. $p \subseteq s$ it holds that $X \cap s \neq \emptyset$.

Note that the disambiguation X of a variable v for a partial state p such that $v \in \mathcal{V}(p)$ can be set simply to $X = \{\langle v, p[v]\rangle\}$, and that the disambiguation X of v for the empty partial state $p = \emptyset$ can be set to $X = \mathcal{F}_v$.

Disambiguations can be used for finding unreachable operators and determining unsolvability of tasks. If, for some operator $o \in \mathcal{O}$, a disambiguation of some $v \in \mathcal{V}$ for $\mathrm{pre}(o)$ is empty, then o is unreachable; and if a disambiguation of some $v \in \mathcal{V}$ for G is empty, the task is unsolvable. So, from now on, we will consider only tasks for which we have non-empty disambiguations of the goal and the operators' preconditions. We use the following **disambiguation maps** $\mathcal{D}$. Given a variable $v \in \mathcal{V}$, $\mathcal{D}(v)$ denotes a disambiguation of v for G. Given an operator $o \in \mathcal{O}$ and $v \in \mathcal{V}(\mathrm{eff}(o))$, $\mathcal{D}(o, v)$ denotes a disambiguation of v for $\mathrm{pre}(o)$.

4 Potential Heuristics

Potential heuristics (Pommerening et al. 2015) were introduced as admissible and consistent heuristics that assign a numerical value (potential) to each fact, and the h-value for a state s is a sum of the potentials of all facts in s. It was shown that potentials can be computed by solving a linear program

(LP) with constraints expressing goal-awareness and consistency of the resulting heuristic. Pommerening, Helmert, and Bonet (2017) extended the concept of potential heuristics to larger sets of facts (higher-dimensional features/conjunctions) by associating each conjunction with a potential, and defining the h-value of a state s as the sum of potentials of the conjunctions true in s. Increasing the size of conjunctions allows to distinguish between more states, eventually leading to a potential heuristic that is optimal.

However, the computation of potentials becomes significantly more difficult as the size of conjunctions grows. Pommerening, Helmert, and Bonet (2017) showed that deciding consistency of a potential heuristic is coNP-hard in general if we consider all conjunctions of size 3 or more. They were however able to identify conditions when the construction of an admissible potential heuristic is tractable, depending on the interactions between the conjunctions. Steinmetz and Hoffmann (2018) showed that admissible higher-dimensional potential heuristics can be computed via a detour to atomic potential heuristics (i.e., potential heuristics over single facts) in the $\Pi^{\mathcal{C}}$ compilation, which we build upon here. The size of $\Pi^{\mathcal{C}}$ grows worst-case exponentially in $|\mathcal{C}|$, yielding an alternative tractability condition—when $\Pi^{\mathcal{C}}$ does not explode.

Since we plan to compute potential heuristics via compilations where each conjunction is explicitly represented as a fact, we formally use only atomic potential heuristics.

Definition 3. A **potential function** is a function $\mathtt{P} : \mathcal{F} \mapsto \mathbb{R}$. A **potential heuristic** for $\mathtt{P}$ maps each state $s \in \mathcal{R}(\Pi)$ to the sum of potentials of facts in s, i.e., $h^{\mathtt{P}}(s) = \sum_{f \in s} \mathtt{P}(f)$.

Potential functions inducing forward admissible potential heuristics can be found by solving LPs, and it was shown by Fišer, Horčík, and Komenda (2020) that, if restricted to the reachable states, potential heuristics can be strengthened by taking disambiguations into account. So, given a disambiguation map $\mathcal{D}$, we can find potential functions $\mathtt{P}$ by solving the following LP: The LP has a variable $\mathtt{P}(f)$ for each fact $f \in \mathcal{F}$, the constraint

$$\sum_{V \in \mathcal{V}} \max_{f \in \mathcal{D}(V)} \mathtt{P}(f) \leq 0$$

ensuring forward goal-awareness, and the constraint

$$\sum_{V \in \mathcal{V}(\mathrm{eff}(o))} \max_{f \in \mathcal{D}(o,V)} \mathtt{P}(f) - \sum_{f \in \mathrm{eff}(o)} \mathtt{P}(f) \leq \mathrm{cost}(o)$$

for each operator $o \in \mathcal{O}$ ensuring forward consistency. Note that the maximization can be easily implemented with auxiliary variables as described by Pommerening et al. (2015). Since the aforementioned constraints ensure f-goal-awareness and f-consistency (and therefore f-admissibility), the objective function of the LP can be freely chosen (Seipp, Pommerening, and Helmert 2015; Fišer, Horčík, and Komenda 2020), e.g., maximization of $\sum_{f \in I} \mathtt{P}(f)$ will result in a potential heuristic with the maximum possible heuristic value for the initial state.

5 $\Pi^{\mathcal{C}}$ Compilation

The first compilation we use for computing potentials over conjunctions is the $\Pi^{\mathcal{C}}$ compilation introduced by Haslum (2012) in the context of strengthening delete-relaxation heuristics. Since then, it proved to be useful in different contexts too (e.g., Keyder, Hoffmann, and Haslum 2014; Fickert, Hoffmann, and Steinmetz 2016; Steinmetz and Hoffmann 2018). In particular, Steinmetz and Hoffmann (2018) already showed that potential heuristics over $\Pi^{\mathcal{C}}$ provide consistent and admissible estimates for the original task. We follow up on their work in that we use the $\Pi^{\mathcal{C}}$ compilation in the FDR formalism (Haslum (2012) used STRIPS), but we also fully utilize mutexes to prune unreachable operators. It was already pointed out by Keyder, Hoffmann, and Haslum (2014) that mutexes are effective in preventing the compilation to blow-up in practice as the size of $\Pi^{\mathcal{C}}$ is worst-case exponential in $|\mathcal{C}|$. Another subtle difference to the work of Steinmetz and Hoffmann is that we consider heuristics defined over reachable states only. For these reasons, we provide not only the description of the compilation, but also full proofs showing that the compilation preserves forward admissibility and forward consistency of heuristics. Moreover, we show that $\Pi^{\mathcal{C}}$ has some disadvantages. Namely, it can induce superfluous paths in the state space.

For the rest of this section, let $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ denote a task with facts $\mathcal{F}$, let $\mathcal{M}$ be a mutex-set, and let $\mathcal{C} \subseteq 2^{\mathcal{F}}$ be a set of conjunctions, each consisting of at least two facts (i.e., $|c| \geq 2$ for every $c \in \mathcal{C}$).

The idea of the $\Pi^{\mathcal{C}}$ compilation is following. First, we encode each conjunction $c \in \mathcal{C}$ as a binary variable v_c: We set v_c to 1 if the conjunction is true in the state s (i.e., $c \subseteq s$), and we set it to 0 if c is false (i.e., $c \not\subseteq s$). Second, we set the initial state and goal so that every v_c has the correct truth value. Finally, we construct multiple operators for each input operator $o \in \mathcal{O}$ so that application of at least one of them maintains the intended values of the v_c variables—this is where the worst-case exponential blow-up comes from as we need to enumerate possible contexts in which each operator can be applied. Before we get to the formal definition of $\Pi^{\mathcal{C}}$, we need to introduce some auxiliary notation that will be helpful in the construction of operators.

Clearly, $o \in \mathcal{O}$ can make $c \in \mathcal{C}$ true or false only if $\mathrm{eff}(o)$ and c share some variables:

$$\mathcal{C}^{\mathrm{eff}}(o) = \{c \in \mathcal{C} \mid \mathcal{V}(c) \cap \mathcal{V}(\mathrm{eff}(o)) \neq \emptyset\}.$$

Nevertheless, we are interested only in conjunctions whose truth values are actually changed by the operator o. Namely, conjunctions $c \in \mathcal{C}^{\mathrm{eff}}(o)$ that are either true before o is applied and false after, or false before and true after. For every $c \in \mathcal{C}$ that is true before applying o it holds that $c \cup \mathrm{pre}(o) \notin \mathcal{M}$ (as $c \cup \mathrm{pre}(o) \in \mathcal{M}$ would imply o is not applicable in a state where c is true). Moreover, o can make c false only if $c \cup \mathrm{post}(o) \in \mathcal{M}$ (as o must affect c by changing at least one variable of c to a different value). Unfortunately, we do not have any means to reliably test whether c is false before the operator's application, but we at least know that c can be made true by o only if $c \cup \mathrm{post}(o) \notin \mathcal{M}$. This leads to the set of conjunctions possibly affected by $o \in \mathcal{O}$ (reduced using mutexes):

$$\begin{aligned}\mathcal{C}^{\mathrm{a}}(o) = &\{c \in \mathcal{C}^{\mathrm{eff}}(o) \mid c \cup \mathrm{pre}(o) \notin \mathcal{M}, c \cup \mathrm{post}(o) \in \mathcal{M}\} \\ &\cup \{c \in \mathcal{C}^{\mathrm{eff}}(o) \mid c \cup \mathrm{post}(o) \notin \mathcal{M}\}.\end{aligned}$$

$\mathcal{V} = \{p, q, r\}, \forall v \in \mathcal{V}\colon \mathrm{dom}(v) = \{0, 1\}$
$I = \{\langle p, 0\rangle, \langle q, 0\rangle, \langle r, 0\rangle\}$
$G = \{\langle q, 1\rangle, \langle r, 1\rangle\}$
$\mathcal{O} = \{o_1, o_2, o_3\}$

o	$\mathrm{pre}(o)$	$\mathrm{eff}(o)$	$\mathrm{post}(o)$
o_1	$\{\langle p, 0\rangle\}$	$\{\langle p, 1\rangle\}$	$\{\langle p, 1\rangle\}$
o_2	$\{\langle p, 0\rangle\}$	$\{\langle r, 1\rangle\}$	$\{\langle p, 0\rangle, \langle r, 1\rangle\}$
o_3	$\{\langle p, 1\rangle\}$	$\{\langle q, 1\rangle\}$	$\{\langle p, 1\rangle, \langle q, 1\rangle\}$

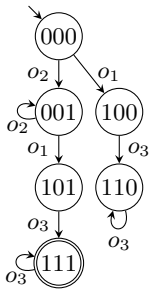

Figure 1: Example planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G\rangle$ and reachable part of its state space on the right where states are annotated with values of variables p, q, r in this order, e.g., the state $\{\langle p, 1\rangle, \langle q, 0\rangle, \langle r, 1\rangle\}$ is annotated as 101.

$\mathcal{C}^{\mathrm{a}}(o)$ can be partitioned into (1) the set of conjunctions always made true by o no matter what was their truth values before applying o:

$$\mathcal{C}^{\mathrm{t}}(o) = \{c \in \mathcal{C}^{\mathrm{a}}(o) \mid c \subseteq \mathrm{post}(o)\},$$

(2) the set of conjunctions always made false by o:

$$\mathcal{C}^{\mathrm{f}}(o) = \{c \in \mathcal{C}^{\mathrm{a}}(o) \mid c \cup \mathrm{post}(o) \in \mathcal{M}\},$$

and (3) the set of conjunctions potentially made true by o depending on the state where o is applied:

$$\mathcal{C}^{\mathrm{p}}(o) = \{c \in \mathcal{C}^{\mathrm{a}}(o) \mid c \cup \mathrm{post}(o) \notin \mathcal{M}, c \not\subseteq \mathrm{post}(o)\}.$$

Lastly, for every subset $X \subseteq \mathcal{C}$, we define the regression of the conjunctions in X over o (i.e., facts from X that if true before the application, are true also after the operator's application):

$$\mathrm{regr}(o, X) = \mathrm{pre}(o) \cup \bigcup_{c \in X} (c \setminus \mathrm{eff}(o)).$$

Now we are ready to formally define the $\Pi^{\mathcal{C}}$ compilation.

Definition 4. Given a planning task Π, a mutex-set $\mathcal{M}$, and a set of conjunctions $\mathcal{C} \subseteq 2^{\mathcal{F}}$ s.t. $|c| \geq 2$ for every $c \in \mathcal{C}$, the planning task $\Pi^{\mathcal{C}} = \langle \mathcal{V}^{\mathcal{C}}, \mathcal{O}^{\mathcal{C}}, I^{\mathcal{C}}, G^{\mathcal{C}}\rangle$ is defined as follows.

(1) $\mathcal{V}^{\mathcal{C}}$ extends $\mathcal{V}$ with a fresh binary variable v_c for each conjunction $c \in \mathcal{C}$, i.e., $\mathcal{V}^{\mathcal{C}} = \mathcal{V} \cup \{v_c \mid c \in \mathcal{C}\}$, and $\mathrm{dom}(v_c) = \{0, 1\}$ for each $c \in \mathcal{C}$.

(2) The initial state is extended with the correct truth values of v_c variables, i.e.,

$$I^{\mathcal{C}} = I \cup \{\langle v_c, 1\rangle \mid c \in \mathcal{C}, c \subseteq I\} \cup \{\langle v_c, 0\rangle \mid c \in \mathcal{C}, c \not\subseteq I\}.$$

(3) The goal is extended with v_c set to 1 whenever c is true in G, i.e., $G^{\mathcal{C}} = G \cup \{\langle v_c, 1\rangle \mid c \in \mathcal{C}, c \subseteq G\}$.

(4) For every operator $o \in \mathcal{O}$ and every subset of conjunctions $X \subseteq \mathcal{C}^p(o)$ potentially made true by o such that (i) $\mathrm{regr}(o, X) \notin \mathcal{M}$, and (ii) X is downward closed on $\mathcal{C}^{\mathrm{p}}(o)$ (i.e., for every $c \in \mathcal{C}^{\mathrm{p}}(o)$ such that there exists $c' \in X$ such that $c \subseteq c'$ it holds that $c \in X$), $\mathcal{O}^{\mathcal{C}}$ has the operator o^X with $\mathrm{cost}(o^X) = \mathrm{cost}(o)$,

$$\begin{aligned}\mathrm{pre}(o^X) &= \mathrm{regr}(o, X)\\ &\quad \cup \{\langle v_c, 1\rangle \mid c \in \mathcal{C}, c \subseteq \mathrm{regr}(o, X)\},\\ \mathrm{eff}(o^X) &= (\mathrm{eff}(o) \cup \{\langle v_c, 1\rangle \mid c \in \mathcal{C}^t(o) \cup X\}\\ &\quad \cup \{\langle v_c, 0\rangle \mid c \in \mathcal{C}^f(o)\}) \setminus \mathrm{pre}(o^X).\end{aligned}$$

$\mathcal{C} = \{c\}, c = \{\langle q, 1\rangle, \langle r, 1\rangle\}$
mutex: $\{\langle p, 0\rangle, \langle q, 1\rangle\}$

o	$\mathcal{C}^{\mathrm{eff}}(o)$	$\mathcal{C}^{\mathrm{a}}(o)$	$\mathcal{C}^{\mathrm{p}}(o)$
o_1	$\emptyset$	$\emptyset$	$\emptyset$
o_2	$\{c\}$	$\emptyset$	$\emptyset$
o_3	$\{c\}$	$\{c\}$	$\{c\}$

$\mathcal{V}^{\mathcal{C}} = \mathcal{V} \cup \{v_c\}, \mathrm{dom}(v_c) = \{0, 1\},$
$I^{\mathcal{C}} = \{\langle p, 0\rangle, \langle q, 0\rangle, \langle r, 0\rangle, \langle v_c, 0\rangle\}$
$G^{\mathcal{C}} = \{\langle q, 1\rangle, \langle r, 1\rangle, \langle v_c, 1\rangle\}$
$\mathcal{O}^{\mathcal{C}} = \{o_1^{\emptyset}, o_2^{\emptyset}, o_3^{\emptyset}, o_3^{\{c\}}\}$
$o_1^{\emptyset} = o_1, o_2^{\emptyset} = o_2, o_3^{\emptyset} = o_3,$
$\mathrm{pre}(o_3^{\{c\}}) = \{\langle p, 1\rangle, \langle r, 1\rangle\}, \mathrm{eff}(o_3^{\{c\}}) = \{\langle q, 1\rangle, \langle v_c, 1\rangle\}$

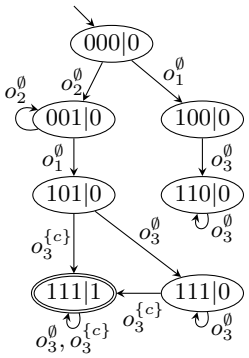

Figure 2: $\Pi^{\mathcal{C}} = \langle \mathcal{V}^{\mathcal{C}}, \mathcal{O}^{\mathcal{C}}, I^{\mathcal{C}}, G^{\mathcal{C}}\rangle$ task constructed from the task Π from Figure 1 and assuming we were able to infer the specified mutex. States in the reachable part of the state space of $\Pi^{\mathcal{C}}$ are annotated with values of variables p, q, r, v_c in this order, e.g., the state $\{\langle p, 1\rangle, \langle q, 0\rangle, \langle r, 1\rangle, \langle v_c, 0\rangle\}$ is annotated as $101|0$.

$\Pi^{\mathcal{C}}$ is well-defined as all variables have finite domains, $I^{\mathcal{C}}$ is defined over all variables, and $\mathrm{pre}(o^X) \cap \mathrm{eff}(o^X) = \emptyset$ for all $o^X \in \mathcal{O}^{\mathcal{C}}$. Note that the blow-up of the compilation comes from enumerating all possible combinations of conjunctions that are potentially made true by an operator (4). It is, however, mitigated by skipping operators that can be proved to be unreachable (4i), and by considering only downward closed sets (4ii). The latter follows from the simple observation that whenever there are $c, c' \in \mathcal{C}$ such that $c' \subseteq c$ and c is true, then c' is also necessarily true and therefore there is no reason for splitting the context where only c is true, only c' is true, and both are true.

Example 5. Consider the task Π from Figure 1, a single conjunction $c = \{\langle q, 1\rangle, \langle r, 1\rangle\}$ ($\mathcal{C} = \{c\}$), and a mutex $\{\langle p, 0\rangle, \langle q, 1\rangle\}$ (and let $\mathcal{M}$ denote the corresponding mutex-set). The resulting $\Pi^{\mathcal{C}}$ task is depicted in Figure 2.

$\mathcal{C}^{\mathrm{eff}}(o_1) = \emptyset$ because o_1 does not affect variables q and r, but $\mathcal{C}^{\mathrm{eff}}(o_2) = \mathcal{C}^{\mathrm{eff}}(o_3) = \{c\}$ because o_2 affects q and o_3 affects r. $\mathcal{C}^{\mathrm{a}}(o_2) = \emptyset$ because $c \cup \mathrm{pre}(o_2) \in \mathcal{M}$ and $c \cup \mathrm{post}(o_2) \in \mathcal{M}$, but $\mathcal{C}^{\mathrm{a}}(o_3) = \{c\}$ because $c \cup \mathrm{post}(o_3) \notin \mathcal{M}$. Furthermore, $\mathcal{C}^{\mathrm{t}}(o_3) = \mathcal{C}^{\mathrm{f}}(o_3) = \emptyset$ and $\mathcal{C}^{\mathrm{p}}(o_3) = \{c\}$ because $c \cup \mathrm{post}(o_3) \notin \mathcal{M}$ and $c \not\subseteq \mathrm{post}(o_3)$. Therefore, we have that $o_1^{\emptyset} = o_1, o_2^{\emptyset} = o_2, o_3^{\emptyset} = o_3$ because $\mathcal{C}^{\mathrm{t}}(o) = \mathcal{C}^{\mathrm{f}}(o) = \emptyset$ for every $o \in \mathcal{O}$; and $\Pi^{\mathcal{C}}$ has one more operator $o_3^{\{c\}}$ such that $\mathrm{pre}(o_3^{\{c\}}) = \{\langle p, 1\rangle, \langle r, 1\rangle\}$ and $\mathrm{eff}(o_3^{\{c\}}) = \{\langle q, 1\rangle, \langle v_c, 1\rangle\}$ because $\mathrm{regr}(o_3, \{c\}) = \{\langle p, 1\rangle, \langle r, 1\rangle\}$ and $c \in \mathcal{C}^{\mathrm{t}}(o_3) \cup \{c\} = \{c\}$.

Note that the reachable part of the state space of $\Pi^{\mathcal{C}}$ differs from Π because it is possible to reach a (non-goal) state $\{\langle p, 1\rangle, \langle q, 1\rangle, \langle r, 1\rangle, \langle v_c, 0\rangle\}$ where v_c has assigned 0 even though c is true in this state. Nevertheless, $\Pi^{\mathcal{C}}$ preserves all plans from Π and does not create any shortcuts.

Given a state s from Π, we define the shorthand

$$\mathcal{C}[s] = s \cup \{\langle v_c, 1\rangle \mid c \in \mathcal{C}, c \subseteq s\} \cup \{\langle v_c, 0\rangle \mid c \in \mathcal{C}, c \not\subseteq s\}$$

that can be used for translating a state s from Π to a state in $\Pi^{\mathcal{C}}$ where all variables v_c have assigned the correct values. Now, we show that every path in $\Pi^{\mathcal{C}}$ corresponds to a path in Π, i.e., the construction of $\Pi^{\mathcal{C}}$ preserves applicability of the original operators from Π, therefore $\Pi^{\mathcal{C}}$ does not introduce any shortcuts into the reachable part of its state space.

Proposition 6. *Let $\pi = \langle o_1^{X_1}, \ldots, o_n^{X_n} \rangle$ denote a path in $\Pi^{\mathcal{C}}$. Then $\pi' = \langle o_1, \ldots, o_n \rangle$ is a path in Π and $\pi'[\![I]\!] = \pi[\![I^{\mathcal{C}}]\!] \cap \mathcal{F}$.*

Proof. Let $s_0, s_1, \ldots, s_n$ denote the intermediate states of π applied on the initial state of $\Pi^{\mathcal{C}}$, i.e., $s_0 = I^{\mathcal{C}}$ and for every $i \in \{1, \ldots, n\}$ it holds that $o_i^{X_i}[\![s_{i-1}]\!] = s_i$. Moreover, let $s'_i = s_i \cap \mathcal{F}$ for every $i \in \{0, 1, \ldots, n\}$. Now, we show that $s'_0 = I$ and every s'_i, $1 \leq i \leq n$, is a state in Π reachable by the sequence of operators $\langle o_1, \ldots, o_i \rangle$.

From the definition of $\Pi^{\mathcal{C}}$ we have that $I^{\mathcal{C}} \cap \mathcal{F} = I = s_0$. Since, for every $i \in \{1, \ldots, n\}$, it holds that $\mathrm{pre}(o_i) \subseteq \mathrm{regr}(o_i, X_i) \subseteq \mathrm{pre}(o_i^{X_i}) \subseteq s_{i-1}$ and $\mathrm{regr}(o_i, X_i) = \mathrm{pre}(o_i^{X_i}) \cap \mathcal{F}$, it follows that $\mathrm{pre}(o_i) \subseteq s'_{i-1}$, i.e., o_i is applicable in s'_{i-1}. Since $\mathrm{eff}(o_i) \setminus \mathrm{regr}(o_i, X_i) = \mathrm{eff}(o_i^{X_i}) \cap \mathcal{F}$ and $\mathrm{regr}(o_i, X_i) \subseteq s_{i-1}$, we have that $o_i^{X_i}[\![s_{i-1}]\!] \cap \mathcal{F} = ((s_{i-1} \setminus \mathcal{F}_{\mathrm{eff}(o_i^{X_i})}) \cup \mathrm{eff}(o_i^{X_i})) \cap \mathcal{F} = (s'_{i-1} \setminus \mathcal{F}_{\mathrm{eff}(o_i)}) \cup \mathrm{eff}(o_i) = o_i[\![s'_{i-1}]\!] = s'_i = s_i \cap \mathcal{F}$. □

Next, we show that for every path π in the original task Π, there is a corresponding path π' in $\Pi^{\mathcal{C}}$ that has exactly the same length and cost, and it leads to a state in $\Pi^{\mathcal{C}}$ with correctly set variables v_c, i.e., $\pi'[\![I^{\mathcal{C}}]\!] = \mathcal{C}[\pi[\![I]\!]]$.

Proposition 7. *Let $\pi = \langle o_1, \ldots, o_n \rangle$ denote a path in Π. Then there exists a path $\pi' = \langle o_1^{X_1}, \ldots, o_n^{X_n} \rangle$ in $\Pi^{\mathcal{C}}$ such that $\pi'[\![I^{\mathcal{C}}]\!] = \mathcal{C}[\pi[\![I]\!]]$.*

Proof. Let $s_0, s_1, \ldots, s_n$ denote the intermediate states of π applied on the initial state of Π, i.e., $s_0 = I$ and for every $i \in \{1, \ldots, n\}$ it holds that $o_i[\![s_{i-1}]\!] = s_i$. Moreover, let $s'_i = \mathcal{C}[s_i]$ for every $i \in \{0, 1, \ldots, n\}$. From the definition of $\Pi^{\mathcal{C}}$ it directly follows that $s'_0 = I^{\mathcal{C}}$.

Assume that there exists a sequence of operators $\pi' = \langle o_1^{X_1}, \ldots, o_{i-1}^{X_{i-1}} \rangle$, for some $i < n$, such that π' is applicable in s'_0 and $\pi'[\![s'_0]\!] = s'_{i-1}$. Now, we show that there exists $o_i^{X_i}$ such that $o_i^{X_i}$ is applicable in s'_{i-1} and $o_i^{X_i}[\![s'_{i-1}]\!] = s'_i$.

Let $X^{\mathrm{f}} = \{c \in \mathcal{C} \mid \langle v_c, 1 \rangle \in s'_{i-1}, \langle v_c, 0 \rangle \in s'_i\}$, $X^{\mathrm{t}} = \{c \in \mathcal{C} \mid \langle v_c, 0 \rangle \in s'_{i-1}, \langle v_c, 1 \rangle \in s'_i\}$, $X^{\mathrm{T}} = \{c \in \mathcal{C} \mid \langle v_c, 1 \rangle \in s'_{i-1}, \langle v_c, 1 \rangle \in s'_i\}$, and $X^{\mathrm{F}} = \{c \in \mathcal{C} \mid \langle v_c, 0 \rangle \in s'_{i-1}, \langle v_c, 0 \rangle \in s'_i\}$. It is easy to see that $X^{\mathrm{f}}, X^{\mathrm{t}}, X^{\mathrm{T}}, X^{\mathrm{F}}$ form a partitioning of $\mathcal{C}$, and $X^{\mathrm{f}} \subseteq \mathcal{C}^{\mathrm{a}}(o_i)$ and $X^{\mathrm{t}} \subseteq \mathcal{C}^{\mathrm{a}}(o_i)$ as any variable assignment can be changed only by the effect of the operator o_i.

Let $X_i = \mathcal{C}^{\mathrm{p}}(o_i) \cap (X^{\mathrm{t}} \cup X^{\mathrm{T}})$. We show that (i) X_i is downward closed on $\mathcal{C}^{\mathrm{p}}(o_i)$ and (ii) $\mathrm{regr}(o_i, X_i) \notin \mathcal{M}$ and (iii) $X^{\mathrm{t}} \setminus \mathcal{C}^{\mathrm{t}}(o_i) \subseteq X_i$ and (iv) $X_i \subseteq X^{\mathrm{t}} \cup X^{\mathrm{T}}$.

(i) X^{t}, X^{T}, and $X^{\mathrm{t}} \cup X^{\mathrm{T}}$ are all downward closed by the construction of s'_i, therefore $\mathcal{C}^{\mathrm{p}}(o_i) \cap (X^{\mathrm{t}} \cup X^{\mathrm{T}})$ is also downward closed.

(ii) For every $c \in X^{\mathrm{t}} \cup X^{\mathrm{T}}$ it holds that $c \subseteq s_i$ by the construction of s'_i, therefore $c \setminus \mathrm{eff}(o_i) \subseteq s_{i-1}$ for every $c \in X^{\mathrm{t}} \cup X^{\mathrm{T}}$, therefore also $\bigcup_{c \in X_i}(c \setminus \mathrm{eff}(o_i)) \subseteq s_{i-1}$, therefore $\mathrm{regr}(o_i, X_i) \notin \mathcal{M}$ because s_{i-1} is reachable.

(iii) It is easy to see that $X^{\mathrm{t}} \subseteq \mathcal{C}^{\mathrm{a}}(o_i)$ and also that $X^{\mathrm{t}} \cap \mathcal{C}^{\mathrm{f}}(o_i) = \emptyset$ because $c \cup \mathrm{post}(o_i) \notin \mathcal{M}$ for every $c \in X^{\mathrm{t}} \cup X^{\mathrm{T}}$. So, since $\mathcal{C}^{\mathrm{t}}(o_i), \mathcal{C}^{\mathrm{f}}(o_i), \mathcal{C}^{\mathrm{p}}(o_i)$ form a partitioning of $\mathcal{C}^{\mathrm{a}}(o_i)$ it follows that $X^{\mathrm{t}} \setminus \mathcal{C}^{\mathrm{t}}(o_i) \subseteq X_i$.

(iv) It follows directly from the construction of X_i.

From (i) and (ii) it follows there exists an operator $o_i^{X_i}$ and since $\mathrm{regr}(o_i, X_i)$ consists of facts unaffected by the operator o_i, it follows that $o_i^{X_i}$ is applicable in s'_{i-1}.

From the construction of $\mathcal{C}^{\mathrm{t}}(o_i)$ it follows that $\mathcal{C}^{\mathrm{t}}(o_i) \subseteq X^{\mathrm{t}}$, and from (iii) and (iv) it follows that $(\mathcal{C}^{\mathrm{t}}(o_i) \cup X_i) \subseteq X^{\mathrm{t}} \cup X^{\mathrm{T}}$. Finally, since $X^{\mathrm{f}} \subseteq \mathcal{C}^{\mathrm{a}}(o_i)$ and for every $c \in X^{\mathrm{f}}$ it holds that $c \cup \mathrm{post}(o_i) \in \mathcal{M}$, we have that $X^{\mathrm{f}} \subseteq \mathcal{C}^{\mathrm{f}}(o_i)$, so it follows that $o_i^{X_i}[\![s'_{i-1}]\!] = s'_i$. □

Now we show that $\Pi^{\mathcal{C}}$ preserves forward consistency and forward admissibility of heuristics. To be precise, we show that if we have a forward admissible (forward consistent) heuristic $h^{\mathcal{C}}$ for $\Pi^{\mathcal{C}}$, then we can cast any state s reachable in Π to another state $\mathcal{C}[s]$ in $\Pi^{\mathcal{C}}$ and use $h^{\mathcal{C}}(\mathcal{C}[s])$ as a forward admissible (forward consistent) estimate for s in Π.

Theorem 8. *If $h^{\mathcal{C}} \colon \mathcal{R}(\Pi^{\mathcal{C}}_{\mathrm{exact}}) \mapsto \mathbb{R}_0^+$ is a forward admissible (forward consistent) heuristic for $\Pi^{\mathcal{C}}$, then $h \colon \mathcal{R}(\Pi) \mapsto \mathbb{R}_0^+$ such that $h(s) = h^{\mathcal{C}}(\mathcal{C}[s])$ for every $s \in \mathcal{R}(\Pi)$ is a forward admissible (forward consistent) heuristic for Π.*

Proof. From Proposition 7 it follows that h is well-defined, because $\mathcal{C}[s] \in \mathcal{R}(\Pi^{\mathcal{C}})$ for every $s \in \mathcal{R}(\Pi)$.

To prove f-admissibility by contradiction, let us assume we have a reachable state $s \in \mathcal{R}(\Pi)$ such that $h(s) > h^\star(s)$. Since $h^{\mathcal{C}}$ is f-admissible in $\Pi^{\mathcal{C}}$, it follows that $h^{\mathcal{C}\star}(s) \geq h^{\mathcal{C}}(\mathcal{C}[s]) = h(s) > h^\star(s)$ where $h^{\mathcal{C}\star}$ denotes an optimal heuristic for $\Pi^{\mathcal{C}}$. Therefore $h^{\mathcal{C}\star}(s) > h^\star(s)$, therefore there exists an I-plan $\langle o_1, \ldots, o_n \rangle$ in Π such that there does not exist any $I^{\mathcal{C}}$-plan $\langle o_1^{X_1}, \ldots, o_n^{X_n} \rangle$ in $\Pi^{\mathcal{C}}$ which is a contradiction with Proposition 7.

To prove f-consistency by contradiction, let us assume we have reachable states $s, s' \in \mathcal{R}(\Pi)$ and an operator $o \in \mathcal{O}$ such that $o[\![s]\!] = s'$ and $h(s) > h(s') + \mathrm{cost}(o)$. From Proposition 7 we have that $\mathcal{C}[s], \mathcal{C}[s'] \in \mathcal{R}(\Pi^{\mathcal{C}})$ and there exists $o^X \in \mathcal{O}^{\mathcal{C}}$ such that $o^X[\![\mathcal{C}[s]]\!] = \mathcal{C}[s']$. Therefore it follows that $h^{\mathcal{C}}(\mathcal{C}[s]) > h^{\mathcal{C}}(\mathcal{C}[s']) + \mathrm{cost}(o)$, and since $h^{\mathcal{C}}$ is f-consistent in $\Pi^{\mathcal{C}}$ we have that $h^{\mathcal{C}}(\mathcal{C}[s']) + \mathrm{cost}(o^X) \geq h^{\mathcal{C}}(\mathcal{C}[s]) > h^{\mathcal{C}}(\mathcal{C}[s']) + \mathrm{cost}(o)$, which is a contradiction because $\mathrm{cost}(o^X) = \mathrm{cost}(o)$ by definition. □

$\Pi^{\mathcal{C}}$ also has some pitfalls. Consider again the task Π and the compilation $\Pi^{\mathcal{C}}$ from Example 5. $\pi = \langle o_1, o_2, o_3 \rangle$ is an I-plan in Π. However, the operator sequence $\pi' = \langle o_1^{\emptyset}, o_2^{\emptyset}, o_3^{\emptyset} \rangle$, while applicable in the initial state, is not an $I^{\mathcal{C}}$-plan in $\Pi^{\mathcal{C}}$. Even though the conjunction c is actually true in the resulting state $\pi'[\![I^{\mathcal{C}}]\!]$, the variable v_c is set to 0 in $\pi'[\![I^{\mathcal{C}}]\!]$. This does not contradict the propositions above, because there still is another sequence of operators $\pi'' \neq \pi'$ from $\Pi^{\mathcal{C}}$ s.t. $\pi''[\![I^{\mathcal{C}}]\!] = \mathcal{C}[\pi[\![s]\!]]$, namely $\pi'' = \langle o_1^{\emptyset}, o_2^{\emptyset}, o_3^{\{c\}} \rangle$.

However, it shows that the construction of $\Pi^{\mathcal{C}}$ can induce superfluous paths in the state space. These paths cannot be shortcuts, but they can be detours or lead to dead-ends.

6 $\Pi^{\mathcal{C}}_{\text{exact}}$ Compilation

Here, we introduce a new compilation $\Pi^{\mathcal{C}}_{\text{exact}}$ that, in contrast to $\Pi^{\mathcal{C}}$, preserves the reachable part of the state space exactly. $\Pi^{\mathcal{C}}_{\text{exact}}$ follows $\Pi^{\mathcal{C}}$ in that it also encodes each conjunction $c \in \mathcal{C}$ as a binary variable v_c, but it encodes operators differently. Each operator o in $\Pi^{\mathcal{C}}_{\text{exact}}$ explicitly encodes truth values of all conjunctions potentially affected by o in both precondition and effect. This can lead to even larger blow-up than in $\Pi^{\mathcal{C}}$, but it allows us to prove that every reachable state s in $\Pi^{\mathcal{C}}_{\text{exact}}$ is of a form $s = \mathcal{C}[s \cap \mathcal{F}]$, i.e., in every reachable state, every v_c is set to 1 whenever $c \subseteq s$, and it is set to 0 otherwise. Therefore, the reachable parts of state spaces of Π and $\Pi^{\mathcal{C}}_{\text{exact}}$ are isomorphic: There is a one-to-one mapping between reachable states in Π and $\Pi^{\mathcal{C}}_{\text{exact}}$, one-to-one mapping between paths preserving costs, and therefore also one-to-one mapping between heuristics. From now on, let $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$, $\mathcal{F}$, $\mathcal{M}$, and $\mathcal{C} \subseteq 2^{\mathcal{F}}$ be as in the previous section. We use some auxiliary notation:

We use the same set $\mathcal{C}^{\text{a}}(o) \subseteq \mathcal{C}$ of conjunctions possibly affected by $o \in \mathcal{O}$. Moreover, we use total functions $f : \mathcal{C}^{\text{a}}(o) \mapsto \{0, 1\}$ for generating possible assignments to variables v_c for all $c \in \mathcal{C}^{\text{a}}(o)$. So, given $o \in \mathcal{O}$ and a total function $f : \mathcal{C}^{\text{a}}(o) \mapsto \{0, 1\}$, we define sets of conjunctions affected by o mapped by f to 1 and 0, respectively:

$$\mathcal{C}^{f,1}(o) = \{c \in \mathcal{C}^{\text{a}}(o) \mid f(c) = 1\},$$

$$\mathcal{C}^{f,0}(o) = \{c \in \mathcal{C}^{\text{a}}(o) \mid f(c) = 0\},$$

and the set of variables from $\mathcal{C}^{f,0}(o)$ that are not part of the precondition or effect of o:

$$\mathcal{V}^{f,0}(o) = \bigcup_{c \in \mathcal{C}^{f,0}(o)} \mathcal{V}(c) \setminus \mathcal{V}(\text{pre}(o) \cup \text{eff}(o)).$$

Moreover, for every partial state p over variables $\mathcal{V}^{f,0}(o)$ (i.e., for every partial state s.t. $\mathcal{V}(p) = \mathcal{V}^{f,0}(o)$), we define

$$\text{expre}(o, f, p) = \text{pre}(o) \cup p \cup \bigcup_{c \in \mathcal{C}^{f,1}(o)} c,$$

$$\text{progr}(o, f, p) = (\text{expre}(o, f, p) \setminus \mathcal{F}_{\text{eff}(o)}) \cup \text{eff}(o).$$

For every operator o, we consider all possible assignments to variables v_c corresponding to possibly affected conjunctions $c \in \mathcal{C}^{\text{a}}(o)$. For each such assignment (a total function $f : \mathcal{C}^{\text{a}}(o) \mapsto \{0, 1\}$), we create a set of operators based on o applicable only in states where the assignment holds. Each of these operators correspond to a possible change of the v_c variables. An operator o changes the value of v_c from 1 to 0 whenever c is true before operator's application and $\text{eff}(o)$ has at least one common variable with c but the values differ. Changing the value of v_c from 0 to 1 is more complicated. Clearly, o can change v_c to 1 only if $\text{eff}(o)$ and c agree on all common variables, but we also need to consider an additional context of assignments to variables from c that are not in $\text{pre}(o)$ or $\text{eff}(o)$ (these are variables $\mathcal{V}^{f,0}(o) \cap \mathcal{V}(c)$). Consider a conjunction $c = \{\langle a, 0\rangle, \langle b, 0\rangle\}$ and an operator o with $\text{pre}(o) = \{\langle a, 1\rangle\}$ and $\text{eff}(o) = \{\langle a, 0\rangle\}$. If c is false in some state s, then either $\langle a, 0\rangle \notin s$ or $\langle b, 0\rangle \notin s$. So, o can make c true only if $\langle b, 0\rangle \in s$, as the effect assigns 0 to the variable a, but the value of b is not specified in $\text{pre}(o)$. In other words, o makes c true depending on the additional context which is not explicitly specified in its precondition. Therefore, the construction iterates over all possible contexts of variables $\mathcal{V}^{f,0}(o)$ so that we are able to exactly determine when the variables v_c change their values from 0 to 1.

So, the blow-up of the $\Pi^{\mathcal{C}}_{\text{exact}}$ compilation is worst-case exponential in $|\mathcal{C}|$ and $|\mathcal{V}|$. We mitigate this blow-up by the application of mutexes to filter out contexts that cannot co-occur with an operator's precondition.

Definition 9. Given a planning task Π, a mutex-set $\mathcal{M}$, and a set of conjunctions $\mathcal{C} \subseteq 2^{\mathcal{F}}$ s.t. $|c| \geq 2$ for every $c \in \mathcal{C}$, the planning task $\Pi^{\mathcal{C}}_{\text{exact}} = \langle \mathcal{V}^{\mathcal{C}}, \mathcal{O}^{\mathcal{C}}_{\text{exact}}, I^{\mathcal{C}}, G^{\mathcal{C}} \rangle$ is defined as follows. $\mathcal{V}^{\mathcal{C}}$, $I^{\mathcal{C}}$, and $G^{\mathcal{C}}$ are exactly the same as for $\Pi^{\mathcal{C}}$, but $\mathcal{O}^{\mathcal{C}}_{\text{exact}}$ is constructed differently: For every operator $o \in \mathcal{O}$ and every total function $f : \mathcal{C}^{\text{a}}(o) \mapsto \{0, 1\}$ and every partial state p over variables $\mathcal{V}^{f,0}(o)$ such that (i) $\text{expre}(o, f, p) \notin \mathcal{M}$ and (ii) for every $c \in \mathcal{C}^{f,0}(o)$ it holds that $c \not\subseteq \text{expre}(o, f, p)$, $\mathcal{O}^{\mathcal{C}}_{\text{exact}}$ has the operator $o^{f,p}$ such that $\text{cost}(o^{f,p}) = \text{cost}(o)$,

$$\text{pre}(o^{f,p}) = \text{expre}(o, f, p) \cup \{\langle v_c, f(c)\rangle \mid c \in \mathcal{C}^{\text{a}}(o)\},$$

$$\begin{aligned}\text{eff}(o^{f,p}) = \text{eff}(o) \setminus \text{pre}(o^{f,p}) \\ \cup \{\langle v_c, 0\rangle \mid c \in \mathcal{C}^{f,1}(o), c \cup \text{post}(o) \in \mathcal{M}\} \\ \cup \{\langle v_c, 1\rangle \mid c \in \mathcal{C}^{f,0}(o), c \subseteq \text{progr}(o, f, p)\}.\end{aligned}$$

As for $\Pi^{\mathcal{C}}$, it is easy to see that $\Pi^{\mathcal{C}}_{\text{exact}}$ is well-defined. Note that the condition (ii) makes sure that we consider only compatible (downward closed) assignments to variables, i.e., whenever we have two conjunctions $c, c' \in \mathcal{C}$ such that $c' \subseteq c$ and $f(v_c) = 1$, then the compilation considers only the functions f where also $f(v_{c'}) = 1$. Also note that effects set v_c to 1 only if v_c is set to 0 in the precondition and vice versa $\langle v_c, 0\rangle \in \text{eff}(o^{f,p})$ only if $\langle v_c, 1\rangle \in \text{pre}(o^{f,p})$.

Example 10. Consider again the task Π, conjunction set $\mathcal{C}$, and mutex-set $\mathcal{M}$ from Example 5. The resulting $\Pi^{\mathcal{C}}_{\text{exact}}$ is depicted in Figure 3 (the values of $\mathcal{C}^{\text{a}}(o)$ are as in Figure 2).

Since $\mathcal{C}^{\text{a}}(o_1) = \mathcal{C}^{\text{a}}(o_2) = \emptyset$ we have that $o_1^{f_\emptyset,\emptyset} = o_1$ and $o_2^{f_\emptyset,\emptyset} = o_2$. For o_3, we need to consider functions f_0 and f_1 such that $f_0(c) = 0$ and $f_1(c) = 1$ because $\mathcal{C}^{\text{a}}(o_3) = \{c\}$. Since $\mathcal{V}^{f_0,0} = \{r\}$ and $\mathcal{V}^{f_1,0} = \emptyset$, we construct 3 variants of the operator o_3: $o_3^{f_0,\{\langle r,0\rangle\}}$, $o_3^{f_0,\{\langle r,1\rangle\}}$, and $o_3^{f_1,\emptyset}$. The effect of $o_3^{f_1,\emptyset}$ is empty because $\text{eff}(o_3) \subset \text{expre}(o_3, f_1, \emptyset)$ and $c \cup \text{post}(o_3) \notin \mathcal{M}$. Moreover, v_c is not set in the effect of $o_3^{f_0,\{\langle r,0\rangle\}}$ because $c \not\subseteq \text{progr}(o_3, f_0, \{\langle r, 0\rangle\})$, i.e., $o_3^{f_0,\{\langle r,0\rangle\}}$ does not make c true. Observe that the reachable parts of the state spaces of $\Pi^{\mathcal{C}}_{\text{exact}}$ and Π are isomorphic, and that the assignment to v_c exactly tracks whether c is true or false in each state (which is not the case for $\Pi^{\mathcal{C}}$).

Analogously to the previous section, we use $\mathcal{C}[s]$ to translate states from Π to $\Pi^{\mathcal{C}}_{\text{exact}}$. Now, we show that every path in $\Pi^{\mathcal{C}}_{\text{exact}}$ has its counterpart with the same cost in Π.

$\mathcal{C} = \{c\}, c = \{\langle q, 1\rangle, \langle r, 1\rangle\}$; mutex: $\{\langle p, 0\rangle, \langle q, 1\rangle\}$

$\mathcal{V}^{\mathcal{C}}, I^{\mathcal{C}}, G^{\mathcal{C}}$ as in $\Pi^{\mathcal{C}}$ (see Figure 2)

$f_\emptyset : \emptyset \mapsto \{0, 1\}$; $f_0, f_1 : \{c\} \mapsto \{0, 1\}$, $f_0(c) = 0$, $f_1(c) = 1$

$\mathrm{expre}(o_3, f_0, \{\langle r, 0\rangle\}) = \{\langle p, 1\rangle, \langle r, 0\rangle\}$
$\mathrm{progr}(o_3, f_0, \{\langle r, 0\rangle\}) = \{\langle p, 1\rangle, \langle q, 1\rangle, \langle r, 0\rangle\}$
$\mathrm{expre}(o_3, f_0, \{\langle r, 1\rangle\}) = \{\langle p, 1\rangle, \langle r, 1\rangle\}$
$\mathrm{progr}(o_3, f_0, \{\langle r, 1\rangle\}) = \{\langle p, 1\rangle, \langle q, 1\rangle, \langle r, 1\rangle\}$
$\mathrm{expre}(o_3, f_1, \emptyset) = \mathrm{progr}(o_3, f_1, \emptyset) = \{\langle p, 1\rangle, \langle q, 1\rangle, \langle r, 1\rangle\}$

$\mathcal{O}^{\mathcal{C}}_{\mathrm{exact}} = \{o_1^{f_\emptyset,\emptyset}, o_2^{f_\emptyset,\emptyset}, o_3^{f_0,\{\langle r,0\rangle\}}, o_3^{f_0,\{\langle r,1\rangle\}}, o_3^{f_1,\emptyset}\}$

o	$\mathrm{pre}(o)$	$\mathrm{eff}(o)$
$o_1^{f_\emptyset,\emptyset}$	$\{\langle p, 0\rangle\}$	$\{\langle p, 1\rangle\}$
$o_2^{f_\emptyset,\emptyset}$	$\{\langle p, 0\rangle\}$	$\{\langle r, 1\rangle\}$
$o_3^{f_0,\{\langle r,0\rangle\}}$	$\{\langle p, 1\rangle, \langle r, 0\rangle, \langle v_c, 0\rangle\}$	$\{\langle q, 1\rangle\}$
$o_3^{f_0,\{\langle r,1\rangle\}}$	$\{\langle p, 1\rangle, \langle r, 1\rangle, \langle v_c, 0\rangle\}$	$\{\langle q, 1\rangle, \langle v_c, 1\rangle\}$
$o_3^{f_1,\emptyset}$	$\{\langle p, 1\rangle, \langle r, 1\rangle, \langle q, 1\rangle, \langle v_c, 1\rangle\}$	$\emptyset$

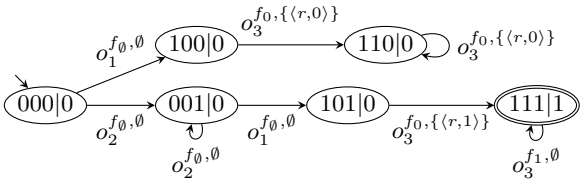

Figure 3: $\Pi^{\mathcal{C}}_{\mathrm{exact}} = \langle \mathcal{V}^{\mathcal{C}}, \mathcal{O}^{\mathcal{C}}_{\mathrm{exact}}, I^{\mathcal{C}}, G^{\mathcal{C}}\rangle$ task constructed from the task Π from Figure 1 (assuming we were able to infer the specified mutex). States in the reachable part of the state space of $\Pi^{\mathcal{C}}_{\mathrm{exact}}$ are annotated as in Figure 2.

Proposition 11. *Let $\pi = \langle o_1^{f_1,p_1}, \ldots, o_n^{f_n,p_n}\rangle$ denote a path in $\Pi^{\mathcal{C}}_{\mathrm{exact}}$. Then $\pi' = \langle o_1, \ldots, o_n\rangle$ is a path in Π and $\pi'[\![I]\!] = \pi[\![I^{\mathcal{C}}]\!] \cap \mathcal{F}$.*

Proof. Let $s_0, s_1, \ldots, s_n$ denote the intermediate states of π applied on the initial state of $\Pi^{\mathcal{C}}_{\mathrm{exact}}$, i.e., $s_0 = I^{\mathcal{C}}$ and for every $i \in \{1, \ldots, n\}$ it holds that $o_i^{f_i,p_i}[\![s_{i-1}]\!] = s_i$. Moreover, let $s'_i = s_i \cap \mathcal{F}$ for every $i \in \{0, 1, \ldots, n\}$. Now, we show that $s'_0 = I$ and, for every $i \in \{1, \ldots, n\}$, s'_i is a state in Π reachable by the sequence of operators $\langle o_1, \ldots, o_i\rangle$, and $s'_i = s_i \cap \mathcal{F}$.

From the definition of $\Pi^{\mathcal{C}}_{\mathrm{exact}}$ we have that $I^{\mathcal{C}} \cap \mathcal{F} = I = s'_0$. Since, for every $i \in \{1, \ldots, n\}$, it holds that $\mathrm{pre}(o_i) \subseteq \mathrm{expre}(o_i, f_i, p_i) \subseteq \mathrm{pre}(o_i^{f_i,p_i}) \subseteq s_{i-1}$, it follows that $\mathrm{pre}(o_i) \subseteq s'_{i-1}$, i.e., o_i is applicable in s'_{i-1}. Since $\mathrm{eff}(o_i^{f_i,p_i}) \cap \mathcal{F} = \mathrm{eff}(o) \setminus \mathrm{pre}(o_i^{f_i,p_i})$ it follows that for every $f \in \mathrm{eff}(o)$ it holds that either $f \in \mathrm{pre}(o_i^{f_i,p_i})$ (and therefore also $f \in s'_{i-1}$ and $f \in s'_i$) or $f \in \mathrm{eff}(o_i^{f_i,p_i})$ (and therefore $f \in s'_i$). Finally, since $\mathrm{eff}(o_i^{f_i,p_i}) \cap \mathcal{F} \subseteq \mathrm{eff}(o_i)$ we have $o_i^{f_i,p_i}[\![s_{i-1}]\!] \cap \mathcal{F} = o_i[\![s'_{i-1}]\!] = s'_i$. □

Next, we show that for every reachable state s in $\Pi^{\mathcal{C}}_{\mathrm{exact}}$, it holds that v_c is set to 1 whenever c is true in s, and v_c is set to 0 whenever c is false in s. In other words, in contrast to $\Pi^{\mathcal{C}}$, all paths in $\Pi^{\mathcal{C}}_{\mathrm{exact}}$ lead to states with correctly set v_c values.

Proposition 12. *Let $s \in \mathcal{R}(\Pi^{\mathcal{C}}_{\mathrm{exact}})$ denote a reachable state in $\Pi^{\mathcal{C}}_{\mathrm{exact}}$. Then $s = \mathcal{C}[s \cap \mathcal{F}]$.*

Proof. Since $I^{\mathcal{C}} = \mathcal{C}[I^{\mathcal{C}} \cap \mathcal{F}]$ by construction, it follows that if there exists a reachable state $s \in \mathcal{R}(\Pi^{\mathcal{C}}_{\mathrm{exact}})$ such that $s \neq \mathcal{C}[s \cap \mathcal{F}]$, then there also exist a reachable state $s' \in \mathcal{R}(\Pi^{\mathcal{C}}_{\mathrm{exact}})$ and an operator $o^{f,p} \in \mathcal{O}^{\mathcal{C}}_{\mathrm{exact}}$ applicable in s' such that $s' = \mathcal{C}[s' \cap \mathcal{F}]$ and $o^{f,p}[\![s']\!] = s$.

We need to investigate four cases:

(i) There exists $c \in \mathcal{C}$ such that $c \subseteq s'$ and $\langle v_c, 1\rangle \in s'$ and $c \subseteq s$ and $\langle v_c, 0\rangle \in s$: The assignment to v_c can change to $\langle v_c, 0\rangle$ only if $c \cup \mathrm{post}(o) \in \mathcal{M}$ which is in contradiction with $c \subseteq s$ because $\mathrm{post}(o) \subseteq s$.

(ii) There exists $c \in \mathcal{C}$ such that $c \subseteq s'$ and $\langle v_c, 1\rangle \in s'$ and $c \not\subseteq s$ and $\langle v_c, 1\rangle \in s$: From $c \subseteq s'$ and $c \not\subseteq s$ we have that $c \cup \mathrm{post}(o) \in \mathcal{M}$, therefore $\langle v_c, 0\rangle \in s$ by construction of $\mathrm{eff}(o^{f,p})$, which is in contradiction with $\langle v_c, 1\rangle \in s$.

(iii) There exists $c \in \mathcal{C}$ such that $c \not\subseteq s'$ and $\langle v_c, 0\rangle \in s'$ and $c \not\subseteq s$ and $\langle v_c, 1\rangle \in s$: The assignment to v_c can change to $\langle v_c, 1\rangle$ only if $c \subseteq \mathrm{progr}(o, f, p)$ which is in contradiction with $c \not\subseteq s$ because $\mathrm{progr}(o, f, p) \subseteq s$.

(iv) There exists $c \in \mathcal{C}$ such that $c \not\subseteq s'$ and $\langle v_c, 0\rangle \in s'$ and $c \subseteq s$ and $\langle v_c, 0\rangle \in s$: From $c \not\subseteq s'$ and $c \subseteq s$ and Proposition 11 it follows that $c \setminus s' \subseteq \mathrm{eff}(o)$, therefore $c \in \mathcal{C}^{\mathrm{a}}(o)$. Therefore it follows from $\langle v_c, 0\rangle \in s'$ that $f(c) = 0$, therefore $p = c \setminus (\mathrm{pre}(o) \cup \mathrm{eff}(o))$, therefore $\mathcal{V}(c \setminus \mathrm{eff}(o)) \subseteq \mathcal{V}(\mathrm{expre}(o, f, p))$, therefore $\mathcal{V}(c) \subseteq \mathcal{V}(\mathrm{progr}(o, f, p))$. Therefore $c \subseteq \mathrm{progr}(o, f, p)$ because $\mathrm{progr}(o, f, p) \subseteq s$, therefore $\langle v_c, 1\rangle \in s$ which contradicts $\langle v_c, 0\rangle \in s$. □

Next, we show that for every path in Π there is a corresponding path in $\Pi^{\mathcal{C}}_{\mathrm{exact}}$ with exactly the same cost.

Proposition 13. *Let $\pi = \langle o_1, \ldots, o_n\rangle$ denote a path in Π. Then there exists a path $\pi' = \langle o_1^{f_1,p_1}, \ldots, o_n^{f_n,p_n}\rangle$ in $\Pi^{\mathcal{C}}_{\mathrm{exact}}$ and $\pi'[\![I^{\mathcal{C}}]\!] = \mathcal{C}[\pi[\![I]\!]]$.*

Proof. Let $s_0, s_1, \ldots, s_n$ denote the intermediate states of π applied on I, i.e., $s_0 = I$ and for every $i \in \{1, \ldots, n\}$ it holds that $o_i[\![s_{i-1}]\!] = s_i$. Moreover, let $s'_i = \mathcal{C}[s_i]$ for every $i \in \{0, 1, \ldots, n\}$. From Definition 9 it follows that $s'_0 = I^{\mathcal{C}}$.

Let us assume that there exists a sequence of operators $\pi' = \langle o_1^{f_1,p_1}, \ldots, o_{i-1}^{f_{i-1},p_{i-1}}\rangle$, for some $i < n$, such that π' is applicable in s'_0 and $\pi'[\![s'_0]\!] = s'_{i-1}$. Now, we show that there exists $o_i^{f_i,p_i}$ such that $o_i^{f_i,p_i}$ is applicable in s'_{i-1} and $o_i^{f_i,p_i}[\![s'_{i-1}]\!] = s'_i$. From Proposition 11 we have that $\mathrm{pre}(o_i) \subseteq s_{i-1} = s'_{i-1} \cap \mathcal{F}$. Let $f_i : \mathcal{C}^{\mathrm{a}}(o) \mapsto \{0, 1\}$ denote a total function such that for every $c \in \mathcal{C}^{\mathrm{a}}(o)$ it holds that $\langle v_c, f(c)\rangle \in s'_{i-1}$, and let $p_i = s'_{i-1} \cap \mathcal{F}_{\mathcal{V}^{f_i,0}(o_i)}$. It is easy to see that f_i is well-defined because $\mathcal{V}(s'_{i-1}) = \mathcal{V}$. It is also easy to see that p_i is a partial state, and since we have that $\mathrm{pre}(o_i) \subseteq s'_{i-1}$ and $p \subseteq s'_{i-1}$ and, for every $c \in \mathcal{C}^{f_i,1}(o_i)$, $c \subseteq s'_{i-1}$, it follows that $\mathrm{expre}(o_i) \notin \mathcal{M}$ and therefore $o_i^{f_i,p_i} \in \mathcal{O}^{\mathcal{C}}_{\mathrm{exact}}$, and for the same reasons we have that $\mathrm{pre}(o_i^{f_i,p_i}) \subseteq s'_{i-1}$. Therefore $o_i^{f_i,p_i}$ is applicable in s'_{i-1} and the rest follows from Proposition 12. □

Finally, we show that there is a one-to-one correspondence between heuristics in Π and $\Pi^{\mathcal{C}}_{\text{exact}}$ preserving forward admissibility and forward consistency (as well as optimality). It is because every path in Π can be mapped to a path in $\Pi^{\mathcal{C}}_{\text{exact}}$ (Proposition 13), every path in $\Pi^{\mathcal{C}}_{\text{exact}}$ can be mapped back to a path in Π (Proposition 11), and there is a bijective mapping between all intermediate states of all paths in Π and $\Pi^{\mathcal{C}}_{\text{exact}}$ (Proposition 12).

Theorem 14. *(A) If $h^{\mathcal{C}}_{\text{ex}}\colon \mathcal{R}(\Pi^{\mathcal{C}}_{\text{exact}}) \mapsto \mathbb{R}^+_0$ is an f-admissible (f-consistent, optimal) heuristic for $\Pi^{\mathcal{C}}_{\text{exact}}$, then $h\colon \mathcal{R}(\Pi) \mapsto \mathbb{R}^+_0$ such that $h(s) = h^{\mathcal{C}}_{\text{ex}}(\mathcal{C}[s])$ for every $s \in \mathcal{R}(\Pi)$ is an f-admissible (f-consistent, optimal) heuristic for Π.*

(B) If $h\colon \mathcal{R}(\Pi) \mapsto \mathbb{R}^+_0$ is an f-admissible (f-consistent, optimal) heuristic for Π, then $h^{\mathcal{C}}_{\text{ex}}\colon \mathcal{R}(\Pi^{\mathcal{C}}_{\text{exact}}) \mapsto \mathbb{R}^+_0$ such that $h^{\mathcal{C}}_{\text{ex}}(s) = h(s \cap \mathcal{F})$ for every $s \in \mathcal{R}(\Pi^{\mathcal{C}}_{\text{exact}})$ is an f-admissible (f-consistent, optimal) heuristic for $\Pi^{\mathcal{C}}_{\text{exact}}$.

Proof. Let $\mathcal{T}(\Pi)$ denote the set of all paths in Π, and let $\mathcal{T}(\Pi^{\mathcal{C}}_{\text{exact}})$ denote the set of all paths in $\Pi^{\mathcal{C}}_{\text{exact}}$. From Proposition 13, it follows there exists a total function $f\colon \mathcal{T}(\Pi) \mapsto \mathcal{T}(\Pi^{\mathcal{C}}_{\text{exact}})$, and from Proposition 11 it follows there exists a total function $g\colon \mathcal{T}(\Pi^{\mathcal{C}}_{\text{exact}}) \mapsto \mathcal{T}(\Pi)$. From Proposition 12 it follows that $f(g(\pi)) = \pi$ for every $\pi \in \mathcal{T}(\Pi^{\mathcal{C}}_{\text{exact}})$ and $g(f(\pi)) = \pi$ for every $\pi \in \mathcal{T}(\Pi)$, therefore f and g are bijective functions and inverses of each other. From the construction of $\Pi^{\mathcal{C}}_{\text{exact}}$ and Propositions 11 and 13 it follows that $\text{cost}(\pi) = \text{cost}(f(\pi))$ for every $\pi \in \mathcal{T}(\Pi)$. Finally, from Proposition 12 it folows there exists a bijective function $e : \mathcal{R}(\Pi) \mapsto \mathcal{R}(\Pi^{\mathcal{C}}_{\text{exact}})$ such that $e(s) = \mathcal{C}[s]$ for every $s \in \mathcal{R}(\Pi)$ and $e^{-1}(s) = s \cap \mathcal{F}$ for every $s \in \mathcal{R}(\Pi^{\mathcal{C}}_{\text{exact}})$, and so (A) and (B) directly follows. □

Despite the encouraging theoretical result formulated in Theorem 14, the construction of $\Pi^{\mathcal{C}}_{\text{exact}}$ can result in having unreachable operators (as the input Π can have unreachable operators too). Therefore, inferring potential heuristics as described in Section 4 does not guarantee we are able to find all possible potential heuristics (as it is not guaranteed in general even for Π).

7 Potential Heuristics over Conjunctions

Using potential heuristics over conjunctions in explicit-state search is straightforward. Given a set of conjunctions $\mathcal{C}$, we construct either the $\Pi^{\mathcal{C}}$ or the $\Pi^{\mathcal{C}}_{\text{exact}}$ compilation of the input planning task Π, compute a potential heuristic $h^{\mathcal{C}}$ in $\Pi^{\mathcal{C}}$ or $\Pi^{\mathcal{C}}_{\text{exact}}$ as described in Section 4, and finally use the heuristic h such that $h(s) = h^{\mathcal{C}}(\mathcal{C}[s])$ for all reachable states $s \in \mathcal{R}(\Pi)$ when running the search in Π. Theorems 8 and 14 show that such a heuristic h is f-admissible and f-consistent.

Moreover, $\Pi^{\mathcal{C}}$ and $\Pi^{\mathcal{C}}_{\text{exact}}$ compilations are ordinary FDR tasks that preserve all optimal plans (and do not induce any shortcuts), therefore we can plan directly in them. Such approach does not offer many benefits for explicit-state search as the larger number of operators makes generating successor states slower, more variables lead to larger memory consumption for storing generated states, and the reachable state space of $\Pi^{\mathcal{C}}$ is possibly larger than the reachable state space of Π (although not in the case of $\Pi^{\mathcal{C}}_{\text{exact}}$). However, there are planning techniques that might benefit from this approach.

Algorithm 1: Inference of improving conjunctions.

Input: A task Π with facts $\mathcal{F}$, an optimization criteria Opt
Output: A set of conjunctions $\mathcal{C}_{\text{o}}$

1 $\mathcal{C}_{\text{o}} \leftarrow \emptyset$; $\mathtt{P} \leftarrow$ potential function for Π maximizing Opt;
2 **for each** $k = 2, \ldots, |\mathcal{V}|$ **do**
3 **for each** $c \subseteq \mathcal{F}, |c| = k, c \notin \mathcal{M}$ **do**
4 **if** *time limit reached* **then return** $\mathcal{C}_{\text{o}}$;
5 $\mathcal{C} \leftarrow \mathcal{C}_{\text{o}} \cup \{x \subseteq c \mid |x| \geq 2\}$;
6 $\mathtt{P}' \leftarrow$ potential func. for $\Pi^{\mathcal{C}}$ (or $\Pi^{\mathcal{C}}_{\text{exact}}$) max. Opt;
7 **if** $\mathtt{P}'$ *is improvement over* $\mathtt{P}$ **then**
8 $\mathcal{C}_{\text{o}} \leftarrow \mathcal{C}$; $\mathtt{P} \leftarrow \mathtt{P}'$;
9 **go to** 2;
10 **return** $\mathcal{C}_{\text{o}}$

Fišer, Torralba, and Hoffmann (2022a,b) showed that potential heuristics can be transformed into operator-potential heuristics associating each operator o with the change of heuristic value of the corresponding potential heuristic induced by o. Operator-potential heuristics significantly improve performance of symbolic search that searches over sets of states (represented as binary decision diagrams (Bryant 1986)) rather than individual states. So, we can run this planning technique directly on $\Pi^{\mathcal{C}}$ and $\Pi^{\mathcal{C}}_{\text{exact}}$ where the increased informativeness of potential heuristics automatically translates into operator-potential heuristics. This does not mean that the symbolic search will not suffer from using larger planning tasks. Also note that different variants of operators o^X in $\Pi^{\mathcal{C}}$ ($o^{f,p}$ in $\Pi^{\mathcal{C}}_{\text{exact}}$) corresponding to the same operator o from Π can induce a different change of heuristic values. Therefore, we cannot directly translate potential heuristics over conjunctions obtained via compilations to operator-potential heuristics in the original task Π.

The next question is how to obtain conjunctions $\mathcal{C}$ improving potential heuristics. As we already indicated before, we do not attempt do design an efficient way to do it here. Instead, we aim at gathering evidence that it is, indeed, common that there are small sets of conjunctions that can significantly improve potential heuristics. So, here we use a simple greedy algorithm (Algorithm 1) that systematically tries to test conjunctions one by one starting from the smallest ones (lines 2 and 3). For each tested conjunction and all its subsets (line 5), a compilation is constructed and the corresponding potential heuristic inferred using the LP described in Section 4 (line 6). Finally, if the potential heuristic is improvement over the current one, we extend the set of conjunctions (lines 7 and 8) and restart the whole process (line 9).

Potential heuristics can be computed by maximizing different optimization criteria—in our experiments we focus on the maximization of the h-value for the initial state only. The improvement of the resulting heuristics can also be measured in different ways. Here, we simply compare objective values of the corresponding LPs—since we maximize, higher values indicate a better heuristic. However, note that the inference using LP cannot guarantee dominance between potential heuristics in a general sense (one upper-bounds the

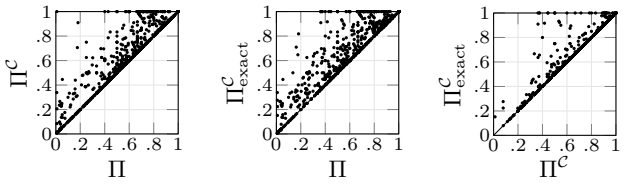

Figure 4: Comparison of h-values for initial states as ratios to the $h^\star$-values for tasks where $h^\star$-values are known. $\Pi^{\mathcal{C}}$ vs. $\Pi^{\mathcal{C}}_{\text{exact}}$ is compared over the same set of conjunctions inferred using $\Pi^{\mathcal{C}}_{\text{exact}}$.

other in *all* reachable states). It can only ensure dominance with respect to the optimization criteria (e.g., h-value for the initial state, or average h-value over all syntactic states).

8 Experimental Evaluation

The proposed method was implemented[1] in C and evaluated on a cluster with Intel Xeon E5-2650v3 processors and 4 GB memory limit for each process. We used all planning domains from the optimal track of International Planning Competitions (IPCs) from 1998 to 2023 excluding the ones containing conditional effects after translation. We merged, for each domain, all benchmarks from different IPCs leaving 54 domains and 1 806 tasks overall. Operators and facts are pruned with the h^2 heuristic in forward and backward direction (Alcázar and Torralba 2015), and the translation from PDDL to FDR uses the mutex groups inference proposed by Fišer (2020, 2023).

For explicit-state search, we ran A* (Hart, Nilsson, and Raphael 1968) with two variants of potential heuristics: `I` denotes maximization of the h-value of the initial state (Pommerening et al. 2015), and `A` denotes maximization for the average (syntactic) state while enforcing the maximum h-value for the initial state (Seipp, Pommerening, and Helmert 2015; Fišer, Horčík, and Komenda 2020). We also evaluated symbolic search with operator-potential heuristics (Fišer, Torralba, and Hoffmann 2022a,b): We used forward search with `A` (denoted by $\overrightarrow{\texttt{A}}$), backward search with `I` ($\overleftarrow{\texttt{I}}$) and the combination of the two in the bidirectional search ($\overrightarrow{\texttt{A}}$-$\overleftarrow{\texttt{I}}$). We use 30 minutes time limit for all search variants (not counting the inference of conjunctions).

Since A* with potential heuristics tends to terminate quickly—either quickly finding a plan or quickly exhausting memory due to the low overhead of the heuristics' evaluation—we also consider simple portfolios P$^{\mathcal{C}}$and P$^{\mathcal{C}}_{\text{ex}}$: We run A* with atomic potential heuristics (`I` or `A`) until it either finds a plan, or runs out of memory. In the latter case, we use the remainder of the 30 minutes time budget by spending 5 minutes on finding conjunctions $\mathcal{C}$, and subsequently re-running A* with potential heuristics over $\mathcal{C}$ using $\Pi^{\mathcal{C}}$ (P$^{\mathcal{C}}$) or $\Pi^{\mathcal{C}}_{\text{exact}}$ (P$^{\mathcal{C}}_{\text{ex}}$). These portfolios count the time spent in finding conjunctions into the whole time limit.

To obtain conjunctions $\mathcal{C}$, we ran Algorithm 1 for 5 minutes for each task, and we set the optimization criteria to the maximization of the h-value for the initial state. Improving conjunctions were found with $\Pi^{\mathcal{C}}$ ($\Pi^{\mathcal{C}}_{\text{exact}}$) in 47 (50) domains and 794 (844) tasks. So, $\Pi^{\mathcal{C}}_{\text{exact}}$ seems to be slightly more successful in this respect. The maximum size of the found conjunction per task ranged from 2 to 8 for $\Pi^{\mathcal{C}}$ and from 2 to 5 for $\Pi^{\mathcal{C}}_{\text{exact}}$. The average (median) was 2.5 (2) for $\Pi^{\mathcal{C}}$ and 2.6 (2) for $\Pi^{\mathcal{C}}_{\text{exact}}$ meaning most of the improvement comes from pairs of facts which is not surprising given Algorithm 1 tests all pairs first. The minimum, maximum, average and median size of the resulting $\mathcal{C}$ was, respectively, 1, 28, 3.9 and 3 for $\Pi^{\mathcal{C}}$, and 1, 38, 3.8 and 3 for $\Pi^{\mathcal{C}}_{\text{exact}}$. This indicates that both compilations used small sets of conjunctions in most cases. Figure 4 compares h-values for initial states as ratios to the optimal heuristic on tasks where we knew $h^\star$ (1 333 tasks). These results show that it is, indeed, possible to obtain significantly more informative potential heuristics via $\Pi^{\mathcal{C}}$ and $\Pi^{\mathcal{C}}_{\text{exact}}$ with just few additional conjunctions.

To get some indication which compilation tends to lead to more informative heuristics, the comparison between $\Pi^{\mathcal{C}}$ and $\Pi^{\mathcal{C}}_{\text{exact}}$ in Figure 4 is done with the same $\mathcal{C}$ (inferred with $\Pi^{\mathcal{C}}_{\text{exact}}$) in each task for both. It seems $\Pi^{\mathcal{C}}_{\text{exact}}$ is more successful than $\Pi^{\mathcal{C}}$ in this respect too. However, $\Pi^{\mathcal{C}}_{\text{exact}}$ also tends to generate larger task representations than $\Pi^{\mathcal{C}}$: The minimum, maximum, average and median ratio between the number of operators in the compilation and in the original planning task was 1, 7.9, 1.2 and 1 for $\Pi^{\mathcal{C}}$, and 1, 46.4, 2.8 and 1.4 for $\Pi^{\mathcal{C}}_{\text{exact}}$. This is also reflected in the speed of computing potential heuristics. We were able to evaluate on average 64.4 conjunctions per second in case of $\Pi^{\mathcal{C}}$ (median was 32.7), and 55.3 with $\Pi^{\mathcal{C}}_{\text{exact}}$ (median 27.6).

Table 1 shows the number of solved tasks (coverage). We are able to increase the coverage of explicit-state search configurations in some domains even with a small set of conjunctions inferred with a simple greedy uninformed algorithm. (Note that A* with `I` over all pairs of facts solves only 451 tasks.) $\Pi^{\mathcal{C}}$ and $\Pi^{\mathcal{C}}_{\text{exact}}$ perform similarly, and the baseline (Π) rarely solves more tasks. The biggest difference between $\Pi^{\mathcal{C}}$ and $\Pi^{\mathcal{C}}_{\text{exact}}$ seems to be in `A` and the woodworking domain where $\Pi^{\mathcal{C}}$ managed to obtain more informed heuristics mainly due to the fact that Algorithm 1 with $\Pi^{\mathcal{C}}$ was able to test larger number of conjunctions within the time limit.

Portfolio results (P$^{\mathcal{C}}$, P$^{\mathcal{C}}_{\text{ex}}$) show that 30 minutes time budget is plenty for A* with potential heuristics to conduct additional analysis to find improving conjunctions. Considering we are typically able to test thousands of different conjunctions within 5 minutes, these results suggest we could obtain better conjunctions by just slightly modifying Algorithm 1 so that it prioritizes more promising conjunctions instead of blindly trying all of them one by one.

The comparison of coverage of symbolic search configurations (Table 1) paints quite different picture from the explicit-state search. Planning directly in $\Pi^{\mathcal{C}}$ or $\Pi^{\mathcal{C}}_{\text{exact}}$ does not seem to have significant effect on $\overrightarrow{\texttt{A}}$ overall although there are few domains where it is beneficial, and it seems to be mostly detrimental for $\overrightarrow{\texttt{A}}$-$\overleftarrow{\texttt{I}}$. The reason is that planning directly in $\Pi^{\mathcal{C}}$ and $\Pi^{\mathcal{C}}_{\text{exact}}$ tends to consume more memory because all binary decision diagrams (BDDs) grow as the task grows, therefore also manipulating BDDs is slower. In

[1] https://gitlab.com/danfis/cpddl, branch icaps24-pot-conj

domain	A* with I					A* with A					Symb. $\overrightarrow{\texttt{A}}$			Symb. $\overleftarrow{\texttt{I}}$			Symb. $\overrightarrow{\texttt{A}}$-$\overleftarrow{\texttt{I}}$		
	Π	$\Pi^{\mathcal{C}}$	$\Pi^{\mathcal{C}}_{\text{ex}}$	$\text{P}^{\mathcal{C}}$	$\text{P}^{\mathcal{C}}_{\text{ex}}$	Π	$\Pi^{\mathcal{C}}$	$\Pi^{\mathcal{C}}_{\text{ex}}$	$\text{P}^{\mathcal{C}}$	$\text{P}^{\mathcal{C}}_{\text{ex}}$	Π	$\Pi^{\mathcal{C}}$	$\Pi^{\mathcal{C}}_{\text{ex}}$	Π	$\Pi^{\mathcal{C}}$	$\Pi^{\mathcal{C}}_{\text{ex}}$	Π	$\Pi^{\mathcal{C}}$	$\Pi^{\mathcal{C}}_{\text{ex}}$
agricola (20)	2	2	2	2	2	2	2	2	2	2	**19**	16	18	**4**	0	1	**18**	12	16
barman (34)	11	11	11	11	11	11	11	11	11	11	16	15	**18**	**4**	3	**4**	**14**	11	12
childsnack (20)	0	0	0	0	0	0	0	0	0	0	2	2	**3**	0	0	0	2	2	**3**
depot (22)	**6**	5	**6**	**6**	**6**	11	11	11	11	11	10	10	**11**	4	4	4	10	10	10
driverlog (20)	8	**9**	**9**	**9**	**9**	13	13	13	13	13	13	13	13	10	**11**	**11**	13	13	13
elevators (50)	28	**31**	**31**	**31**	**31**	28	**35**	33	**35**	33	35	**39**	37	10	12	**14**	**41**	34	33
floortile (40)	16	16	16	16	16	16	16	16	16	16	17	16	**19**	26	25	**27**	26	22	**27**
freecell (80)	48	**49**	47	**49**	48	70	70	70	70	70	68	68	68	27	**28**	27	**67**	**67**	66
ged (20)	15	**19**	**19**	**19**	**19**	15	**19**	**19**	**19**	**19**	15	15	15	10	10	**13**	**19**	**19**	15
logistics (63)	12	**17**	**17**	**17**	**17**	21	**25**	**25**	**25**	**25**	28	28	28	19	**24**	23	28	28	28
nomystery (20)	9	**10**	9	**10**	9	14	14	14	14	14	18	18	18	**14**	13	**14**	19	19	19
openstacks (100)	**54**	53	53	**54**	**54**	**54**	53	53	**54**	**54**	**90**	89	86	73	**75**	73	**89**	88	86
parking (40)	0	**1**	**1**	**1**	**1**	14	14	**15**	14	**15**	13	13	13	6	6	6	12	12	**13**
pegsol (50)	48	48	48	48	48	48	48	48	48	48	48	48	48	30	31	**34**	**48**	**48**	46
petri-net-align (20)	10	10	10	10	10	11	11	11	11	11	**11**	**11**	10	1	1	1	**8**	6	5
pipesw-notank (50)	24	24	24	**25**	24	26	26	**27**	26	**27**	24	24	24	9	9	**10**	24	24	24
pipesw-tank (50)	15	**16**	15	**16**	15	17	**18**	17	**18**	17	**21**	20	18	7	7	7	**20**	**20**	19
quantum-layout (20)	12	12	12	12	12	13	13	13	13	13	14	14	**15**	13	13	13	14	14	**15**
ricochet-robots (20)	7	7	7	7	7	7	7	7	7	7	**2**	1	1	0	0	0	**2**	1	**2**
rovers (40)	6	6	**7**	6	**7**	8	8	8	8	8	**14**	13	**14**	**10**	9	**10**	14	14	14
slitherlink (20)	**4**	**4**	3	**4**	**4**	4	4	4	4	4	5	**6**	5	0	0	0	5	**6**	5
snake (20)	14	14	14	14	14	15	15	15	15	15	8	**11**	**11**	0	0	0	**8**	7	7
sokoban (50)	48	48	48	48	48	50	50	50	50	50	50	50	50	**38**	37	**38**	50	50	50
spider (20)	**14**	13	**14**	**14**	**14**	15	15	15	15	15	13	13	13	0	0	0	12	12	12
termes (20)	10	10	10	10	10	12	12	12	12	12	**12**	10	11	7	**13**	**13**	13	13	13
tidybot (40)	**32**	**32**	30	**32**	**32**	32	32	32	32	32	**34**	**34**	32	8	8	**10**	30	30	**32**
tpp (30)	6	**8**	**8**	**8**	**8**	8	8	8	8	8	12	12	12	**8**	7	**8**	12	12	12
visitall (40)	23	23	**24**	23	**24**	29	29	29	29	29	22	22	22	18	18	**19**	22	22	**23**
woodworking (50)	17	**19**	**19**	**19**	**19**	27	**39**	31	**39**	31	46	**48**	44	34	**46**	40	46	**49**	48
others (737)	393	393	393	393	393	429	429	429	429	429	456	456	456	342	342	342	437	437	437
Σ (1806)	892	910	907	**914**	912	1 020	1 047	1 038	**1 048**	1 039	**1 136**	1 135	1 133	732	752	**762**	**1 123**	1 102	1 105

Table 1: Coverage for explicit-state and symbolic search.

some tasks, there is also an issue with the ordering of BDD variables (corresponding to facts in the task) causing significant blow-up of the BDD encodings. This, unfortunately, makes it harder to apply potential heuristics over conjunction in this setting, because we would need to consider conjunctions that not only improve the informativeness of the heuristics, but also induce a good ordering of BDD variables.

Nevertheless, at least the symbolic backward search $\overleftarrow{\texttt{I}}$ seems to benefit from the compilations (more often from $\Pi^{\mathcal{C}}_{\text{exact}}$ than from $\Pi^{\mathcal{C}}$). The main reason is that the backward search usually generates a smaller number of BDDs representing sets of states than the forward direction, but they typically encapsulate larger number of states. So, it seems having more informative heuristics is able to push this method a little bit further, but enough to see a difference in coverage.

9 Conclusion and Future Work

In this work, we focus on higher-dimensional potential heuristics computed via a small number of conjunctions explicitly represented as facts in compilations of the input planning tasks. We propose to use two variants of compilations: a well known $\Pi^{\mathcal{C}}$ compilation (Haslum 2012), and a newly introduced $\Pi^{\mathcal{C}}_{\text{exact}}$ compilation. We compute potential heuristics over conjunctions as atomic potential heuristics in the compilations. Instead of trying to figure out how to find "good" conjunctions guaranteeing increased informativeness of the resulting heuristics, we focus on a basic question whether heuristic estimates can be improved with just few conjunctions and without significantly increasing computational cost in practice. To this end, we use a very simple greedy algorithm that blindly tries small conjunctions one by one, accepting the conjunction if its addition increases h-value for the initial state. We conclude that even in this simple setting, we can, indeed, increase informativeness of potential heuristics with just few conjunctions. Moreover, we provide a machinery for computing potential heuristics over conjunctions requiring only conjunctions to be plugged-in.

We leave many questions unanswered for future work. The first one is how to determine which conjunctions lead to an improvement. Possible directions for answering this question might be previous works on the selection of improving conjunctions for delete-relaxed heuristics (Fickert and Hoffmann 2017) or for learning no-goods using state-equation heuristics (Steinmetz and Hoffmann 2018). Selection of patterns for pattern databases might also be relevant (Edelkamp 2006; Haslum et al. 2007; Franco et al. 2017; Rovner, Sievers, and Helmert 2019). Another question is the exact relationship between $\Pi^{\mathcal{C}}$ and $\Pi^{\mathcal{C}}_{\text{exact}}$ compilations with respect to the possible potential heuristics that they can express via linear programs (LPs). It would also be interesting to know their relationship to the direct encoding in LP described by Pommerening, Helmert, and Bonet (2017). Lastly, it is not entirely clear how to successfully transfer the increased informativeness of potential heuristics to operator-potential heuristics in the context of symbolic search (Fišer, Torralba, and Hoffmann 2022a,b).

Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—project number 389792660—TRR 248 (see https://perspicuous-computing.science), and the European Union's Horizon Europe Research and Innovation program under the grant agreement TUPLES No 101070149.

References

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 2–6.

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS^+ Planning. *Computational Intelligence*, 11(4): 625–655.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1–2): 5–33.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8): 677–691.

Corrêa, A. B.; and Pommerening, F. 2019. An Empirical Study of Perfect Potential Heuristics. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, 114–118.

Edelkamp, S. 2006. Automated Creation of Pattern Database Search Heuristics. In *Proceedings of the 4th Workshop on Model Checking and Artificial Intelligence (MoChArt 2006)*, 35–50.

Fickert, M.; and Hoffmann, J. 2017. Ranking Conjunctions for Partial Delete Relaxation Heuristics in Planning. In *Proceedings of the 10th Annual Symposium on Combinatorial Search (SOCS'17)*.

Fickert, M.; Hoffmann, J.; and Steinmetz, M. 2016. Combining the Delete Relaxation with Critical-Path Heuristics: A Direct Characterization. *Journal of Artificial Intelligence Research*, 56(1): 269–327.

Fišer, D.; and Komenda, A. 2018. Fact-Alternating Mutex Groups for Classical Planning. *Journal of Artificial Intelligence Research*, 61: 475–521.

Fišer, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, 9835–9842.

Fišer, D. 2023. Operator Pruning Using Lifted Mutex Groups via Compilation on Lifted Level. In *Proceedings of the 33nd International Conference on Automated Planning and Scheduling (ICAPS'23)*, 118–127.

Fišer, D.; Horčík, R.; and Komenda, A. 2020. Strengthening Potential Heuristics with Mutexes and Disambiguations. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, 124–133.

Fišer, D.; Torralba, Á.; and Hoffmann, J. 2022a. Operator-Potential Heuristics for Symbolic Search. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*, 9750–9757.

Fišer, D.; Torralba, Á.; and Hoffmann, J. 2022b. Operator-Potentials in Symbolic Search: From Forward to Bidirectional Search. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22)*, 80–89.

Franco, S.; Torralba, A.; Lelis, L. H.; and Barley, M. 2017. On Creating Complementary Pattern Databases. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, 4302–4309.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Haslum, P. 2012. Incremental Lower Bounds for Additive Cost Planning Problems. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 74–82.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07)*, 1007–1012.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *Artificial Intelligence*, 173: 503–535.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2014. Improving Delete Relaxation Heuristics Through Explicitly Represented Conjunctions. *Journal of Artificial Intelligence Research*, 50: 487–533.

Pommerening, F.; Helmert, M.; and Bonet, B. 2017. Higher-Dimensional Potential Heuristics for Optimal Classical Planning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, 3636–3643.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, 3335–3341.

Rovner, A.; Sievers, S.; and Helmert, M. 2019. Counterexample-Guided Abstraction Refinement for Pattern Selection in Optimal Classical Planning. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, 362–367.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New Optimization Functions for Potential Heuristics. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 193–201.

Steinmetz, M.; and Hoffmann, J. 2018. LP Heuristics over Conjunctions: Compilation, Convergence, Nogood Learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, 4837–4843.